What is claimed is:

[Claim 1] 1. A method for refactoring a plurality of interdependent software modules that reside in separate projects, the method comprising:

in response to a change that affects a particular symbol of a software module that resides in a first project, refactoring the software module of the first project to propagate the change to all instances of the particular symbol in the software module;

during the refactoring of the software module of the first project, recording meta data about the refactoring that is required to effect the change; and

automatically propagating the change to a dependent software module residing in a second project, by refactoring the dependent software module based on the recorded meta data about the refactoring that occurred to the software module of the first project.

[Claim 2] 2. The method of claim 1, wherein the dependent software module is refactored with assistance of a symbol table used for resolving symbol references, and wherein said automatically propagating step includes:

copying symbol information about the particular symbol used for the software module of the first project into the symbol table used for refactoring the dependent software module.

[Claim 3] 3. The method of claim 2, further comprising:

removing symbol information about the particular symbol used for the software module of the first project from the symbol table used for refactoring the dependent software module after the refactoring of the dependent software module.

[Claim 4] 4. The method of claim 2, wherein said copying symbol information step includes copying symbol information into a compiler symbol table used for refactoring the dependent software module.

[Claim 5] 5. The method of claim 2, wherein said copying symbol information step includes creating source code based on the recorded meta data.

[Claim 6] 6. The method of claim 5, further comprising:

parsing the source code so as to indirectly inject symbol information into the symbol table used for refactoring the dependent software module.

[Claim 7] 7. The method of claim 5, wherein said copying symbol information step includes indirectly injecting symbol information for a class into the symbol table by parsing the source code and directly injecting symbol information for members of the class into the symbol table.

[Claim 8] 8. The method of claim 5, further comprising:

deleting the source code after refactoring the dependent software module.

[Claim 9] 9. The method of claim 1, wherein the refactoring of the software module of the first project comprises a selected one of renaming a package, changing a method signature, renaming a method, renaming a field, and renaming a class.

[Claim 10] 10. The method of claim 1, wherein said automatically propagating step includes propagating changes to the dependent software module so as to maintain compatibility with the software module of the first project.

[Claim 11] 11. The method of claim 1, wherein said recording step includes recording meta data before and after refactoring of the software module of the first project.

[Claim 12] 12. The method of claim 1, wherein said recording step includes recording information in Extensible Markup Language (XML) format.

[Claim 13] 13. The method of claim 1, wherein the software module of the first project is in a first programming language and the dependent software module of the second project is in a second programming language.

[Claim 14] 14. A computer-readable medium having processor-executable instructions for performing the method of claim 1.

[Claim 15] 15. A downloadable set of processor-executable instructions for performing the method of claim 1.

[Claim 16] 16. A system for automatically applying a refactoring to a second software module based on a refactoring of a first software module, the system comprising:

a recording module for recording information about changes made to the first software module during a refactoring of the first software module;
an injector module for copying symbol information about at least one symbol of the first software module into a symbol table for the second software module; and
a refactoring module for automatically applying a refactoring to the second software module using said symbol table and the recorded information about changes made to the first software module.

[Claim 17] 17. The system of claim 16, wherein the second software module is dependent upon the first software module.

[Claim 18] 18. The system of claim 17, wherein the refactoring module applies changes to the second software module so as to maintain compatibility of the second software module with the first software module.

[Claim 19] 19. The system of claim 16, wherein the refactoring of the first software module comprises a selected one of renaming a package, changing a method signature, renaming a method, renaming a field, and renaming a class.

[Claim 20] 20. The system of claim 16, wherein the injector module creates source code based on the first software module and the recorded information.

[Claim 21] 21. The system of claim 20, wherein said injector module decompiles at least a portion of the first software module for creating source code.

[Claim 22] 22. The system of claim 20, wherein the injector module parses the source code so as to indirectly inject symbol information into said symbol table.

[Claim 23] 23. The system of claim 16, wherein the injector module copies symbol information into a compiler symbol table used by the refactoring module in the refactoring of the second software module.

[Claim 24] 24. The system of claim 16, wherein the first software module comprises a library.

[Claim 25] 25. The system of claim 24, wherein the second software module comprises an application using the library.

[Claim 26] 26. The system of claim 16, wherein said recording module records information before and after refactoring of the first software module.

[Claim 27] 27. The system of claim 16, wherein said recording module records information in Extensible Markup Language (XML) format.

[Claim 28] 28. The system of claim 16, wherein the first software module runs on a first machine and the second software module runs on a second machine.

[Claim 29] 29. The system of claim 16, wherein the first software module is in a first project and the second software module is in a second project.

[Claim 30] 30. The system of claim 16, wherein the first software module is in a first programming language and the second software module is in a second programming language.

[Claim 31] 31. A method for asynchronous refactoring of a plurality of interdependent software programs, the method comprising:

refactoring a first software program so as to change symbols of the first software program;

recording information about changes made to symbols of the first software program during the refactoring of the first software program; and

subsequently, applying the refactoring to a second software program dependent upon the first software program by automatically propagating

changes to symbols of the second software program based on said recorded information.

[Claim 32] 32. The method of claim 31, wherein the second software program is refactored with assistance of a symbol table used for resolving symbol references, and wherein said applying step includes:

copying information about at least one symbol used in the first software program into the symbol table used for refactoring the second software program.

[Claim 33] 33. The method of claim 32, further comprising:

removing said information about at least one symbol used in the first software program from the symbol table after applying the refactoring of the second software program.

[Claim 34] 34. The method of claim 32, wherein said symbol table comprises a compiler symbol table.

[Claim 35] 35. The method of claim 32, wherein said copying information step includes creating source code based on the recorded information and parsing the source code so as to indirectly inject symbol information into the symbol table used for refactoring the second software program.

[Claim 36] 36. The method of claim 35, wherein said copying information step includes indirectly injecting symbol information for a class into the symbol table by parsing the source code and directly injecting symbol information for members of the class into the symbol table.

[Claim 37] 37. The method of claim 32, wherein said applying step includes applying the refactoring with assistance of a compiler for identifying and changing symbols of the second software program.

[Claim 38] 38. The method of claim 31, wherein the refactoring of the first software program comprises a selected one of renaming a package, changing a method signature, renaming a method, renaming a field, and renaming a class.

[Claim 39] 39. The method of claim 31, wherein the first software program comprises a library.

[Claim 40] 40. The method of claim 39, wherein the second software program comprises an application using the library.

[Claim 41] 41. The method of claim 31, wherein said recording step includes recording information before and after refactoring of the first software program.

[Claim 42] 42. The method of claim 31, wherein said recording step includes recording information in Extensible Markup Language (XML) format.

[Claim 43] 43. A computer-readable medium having processor-executable instructions for performing the method of claim 31.

[Claim 44] 44. A downloadable set of processor-executable instructions for performing the method of claim 31.

[Claim 45] 45. A method for applying a refactoring to a plurality of software modules, the method comprising:

recording information about changes made to a first software module during a refactoring of the first software module;
creating at least one symbol table entry based upon the recorded information about changes made to the first software module;
injecting said at least one symbol table entry into a symbol table for a second software module; and
refactoring the second software module using said symbol table and the recorded information about changes made to the first software module.

[Claim 46] 46. The method of claim 45, wherein the second software module is dependent upon the first software module.

[Claim 47] 47. The method of claim 46, wherein said refactoring the second software module step includes applying changes to the second software module so as to maintain compatibility with the first software module.

[Claim 48] 48. The method of claim 45, wherein the refactoring of the first software module comprises a selected one of renaming a package, changing a method signature, renaming a method, renaming a field, and renaming a class.

[Claim 49] 49. The method of claim 45, wherein said creating step includes creating source code based on the first software module and the recorded information.

[Claim 50] 50. The method of claim 49, wherein said step of creating source code includes decompiling at least a portion of the first software module.

[Claim 51] 51. The method of claim 49, wherein said injecting step includes parsing the source code so as to indirectly inject a symbol table entry into said symbol table.

[Claim 52] 52. The method of claim 51, wherein said injecting step includes indirectly injecting symbol table entries for a class into the symbol table by parsing the source code and directly injecting symbol table entries for members of the class into the symbol table.

[Claim 53] 53. The method of claim 45, further comprising:

removing said at least one symbol table entry from said symbol table after applying the refactoring to the second software module.

[Claim 54] 54. The method of claim 45, wherein said symbol table comprises a compiler symbol table.

[Claim 55] 55. The method of claim 45, wherein said recording step includes recording information before and after refactoring of the first software module.

[Claim 56] 56. The method of claim 45, wherein said recording step includes recording information in Extensible Markup Language (XML) format.

[Claim 57] 57. The method of claim 45, wherein the first software module runs on a first machine and the second software module runs on a second machine.

[Claim 58] 58. The method of claim 45, wherein the first software module is in a first programming language and the second software module is in a second programming language.

[Claim 59] 59. A computer-readable medium having processor-executable instructions for performing the method of claim 45.

[Claim 60] 60. A downloadable set of processor-executable instructions for performing the method of claim 45.